An Algorithmic Approach to Recoverable-robust Timetabling on Trees

Work partially supported by the Future and Emerging Technologies Unit of EC (IST priority - 6th FP), under contract no. FP6-021235-2 (project ARRIVAL)

Gianlorenzo D'Angelo[1] Gabriele Di Stefano[1]
Alfredo Navarra[2] Cristina M. Pinotti[2]

[1]Dept. of Electrical and Information Engineering, University of L'Aquila, Italy
{gianlorenzo.dangelo,gabriele.distefano}@univaq.it

[2]Dept. of Mathematics and Informatics, University of Perugia, Italy
{navarra, pinotti}@dmi.unipg.it

Timetabling

Schedule the departure and arrival time of trains in order to reduce the traveling time for passengers

Delay Management

Modify the timetable when unpredictable events cause delays

Recoverable Robustness

Design the timetable in order to easily recover when delays occur [Liebchen et al. 2007, Cicerone et al. 2007]

This work

Keep on study the Recoverable Robustness approach for timetabling

Results of the paper

- We study the Recoverable Robustness Timetabling problem in restricted topologies (Tree)
- We prove that the Problem is *NP*-hard for trees
- We give a pseudo-polynomial time algorithm
- We experimentally evaluate the algorithm on both real data (Italian railways) and randomly generated data

Outline

Outline

The timetabling problem [Schöbel, 2007]

Scheduling the departure and arrival time of trains

Instances

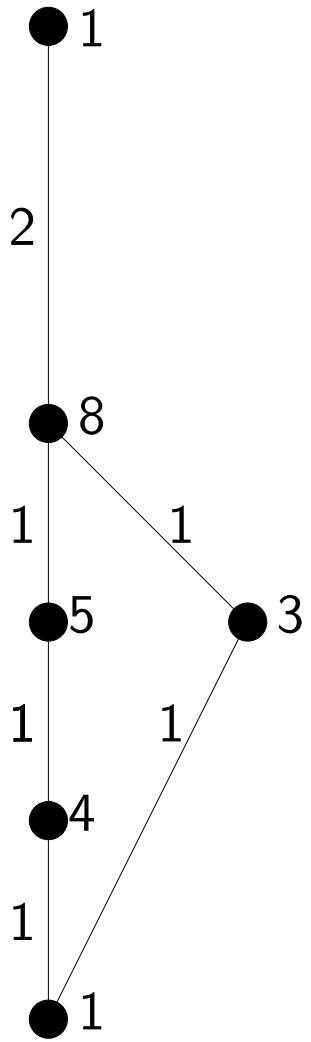
- A *event activity network* $\mathcal{N} = (\mathcal{E}, \mathcal{A})$ made of departure and arrival events $\mathcal{E}$ and activities $\mathcal{A}$
- The *minimum time* $L(a)$ needed for each activity $a \in \mathcal{A}$
- The *number of passengers* $w(v)$ involved in each event $v \in \mathcal{E}$

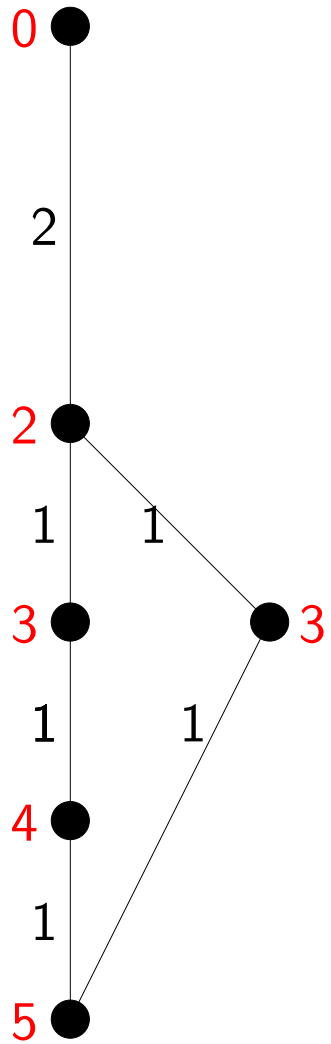
Solutions

A scheduled time $\Pi(v)$ for each event $v \in \mathcal{E}$ in the network, such that Π satisfies the minimum duration time of each activity

Objective

Minimizing the overall traveling time for passengers
$\min f = \sum_{v \in \mathcal{E}} w(v)\Pi(v)$


1
2
8
1
1
5
3
1
1
4
1
1


0
2
2
1
1
3
3
1
1
4
1
5

Timetabling problem TT

$$\min f = \sum_{v \in \mathcal{E}} w(v)\Pi(v)$$

subject to

$$\Pi(v) - \Pi(u) \geq L(a), \qquad \text{for each } a = (u, v) \in \mathcal{A}$$
$$\Pi(v) \in \mathbb{N}, \qquad \text{for each } v \in \mathcal{E}$$

- I: set of instances $i = (\mathcal{N}, L, w)$
- $F(i)$: set of feasible solutions Π for $i \in I$

Modification function M

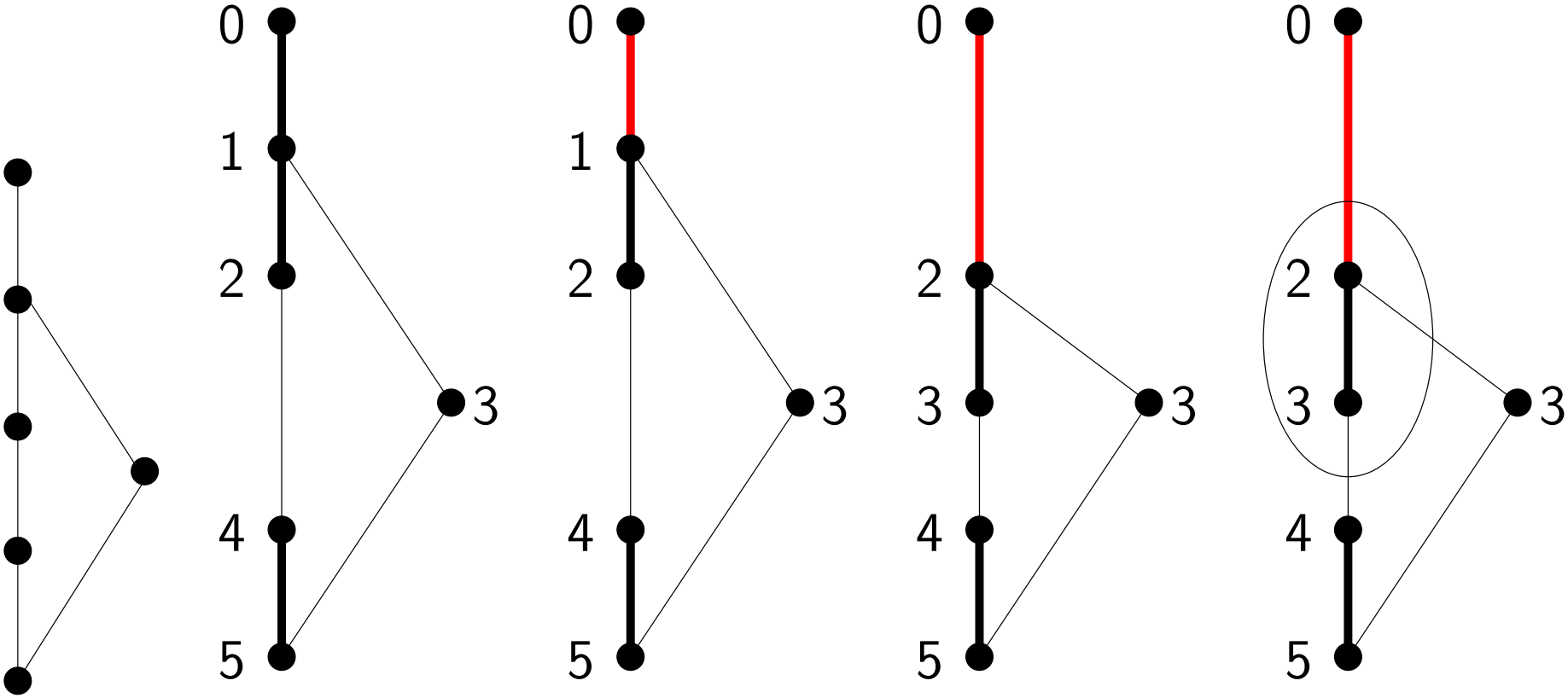
We allow only one delay on an activity a of at most α time
We can model it as an increase of the minimal duration time a
Given $i = (\mathcal{N}, L, w)$,

$$M(i) = \{(\mathcal{N}, L', w) : L' \text{ differs from } L \text{ by at most one activity } \}$$

Recovery capabilities $\mathbb{A}$

A recovery algorithm can change the time of at most Δ *affected events*

$$L(a) = 1 \ \forall a \in \mathcal{A}$$



The Recoverable Robust Timetabling Problem $\mathcal{RTT}$

$\mathcal{RTT} = (TT, M, \mathbb{A})$ is the problem of finding a timetable that can be recovered by changing the time of at most Δ events when a delay of at most α time occurs

Robust Algorithm

A *robust algorithm* for TT is any algorithm A_r which solves $\mathcal{RTT}$.

Price of robustness

The worst case ratio between the cost of the solution computed by A_r and the optimal one is called *price of robustness of* A_r.

$$Prob(\mathcal{RTT}, A_r) = \max_{i \in I} \left\{ \frac{f(A_r(i))}{\min\{f(x) : x \in F(i)\}} \right\}$$

The minimum price of robustness among all the robust algorithms is called *price of robustness of problem*

Previous Works

- $\mathcal{RTT}$ is NP-hard for general event activity networks and directed acyclic graphs
- Linear time approximation algorithm
- Polynomially solvable for linear graphs

In this paper

- We restrict the topology of event activity networks to trees

Outline

Theorem

Finding a solution for $\mathcal{RTT}$ that minimizes f is NP-hard even on tree topologies.

Corollary

Computing $P_{rob}(\mathcal{RTT})$ is NP-hard even on tree topologies.

Outline

Slack times

According to a timetable Π, the slack time $s(a)$ of an activity $a = (u, v)$ is the amount of time assigned to an activity in addiction to its minimum time needed $L(a)$

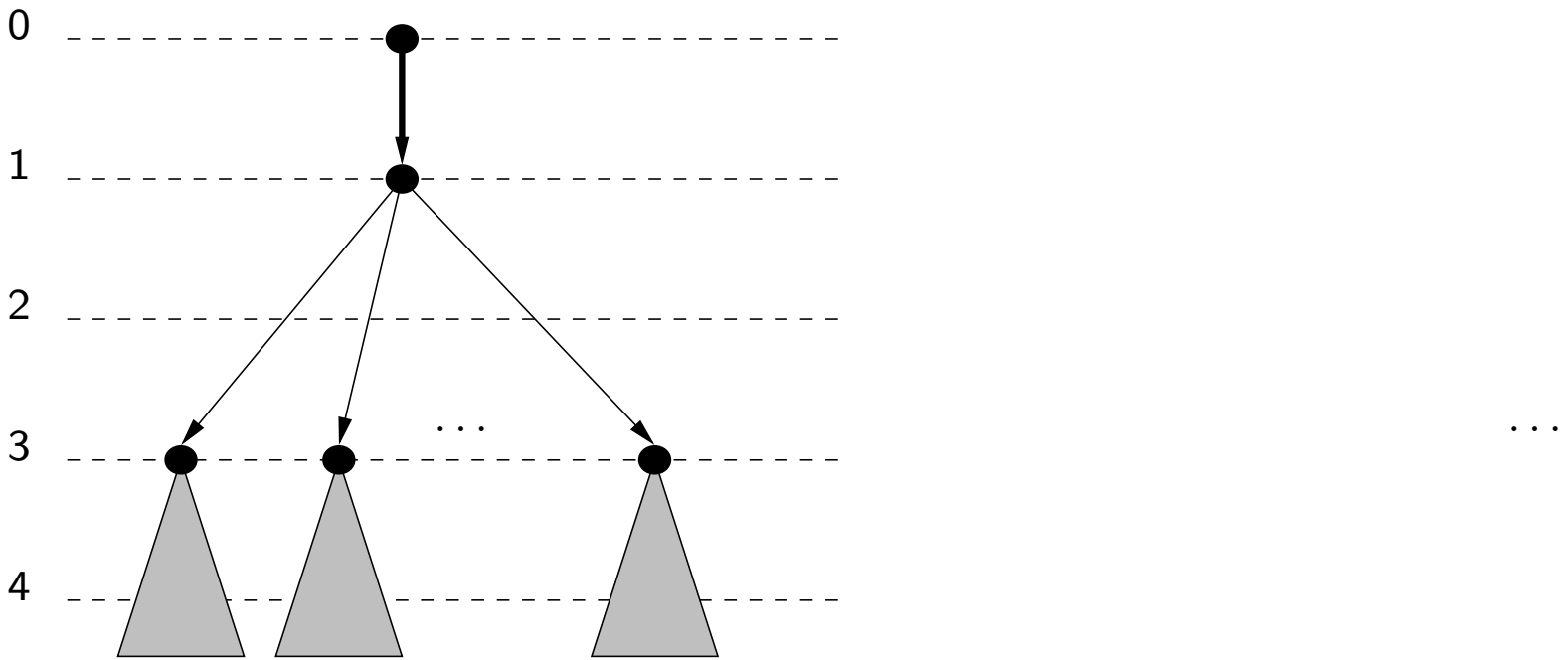
$$s(a) = \Pi(v) - \Pi(u) - L(a)$$

A robust timetable assigns at one least slack time of α every Δ subsequent events

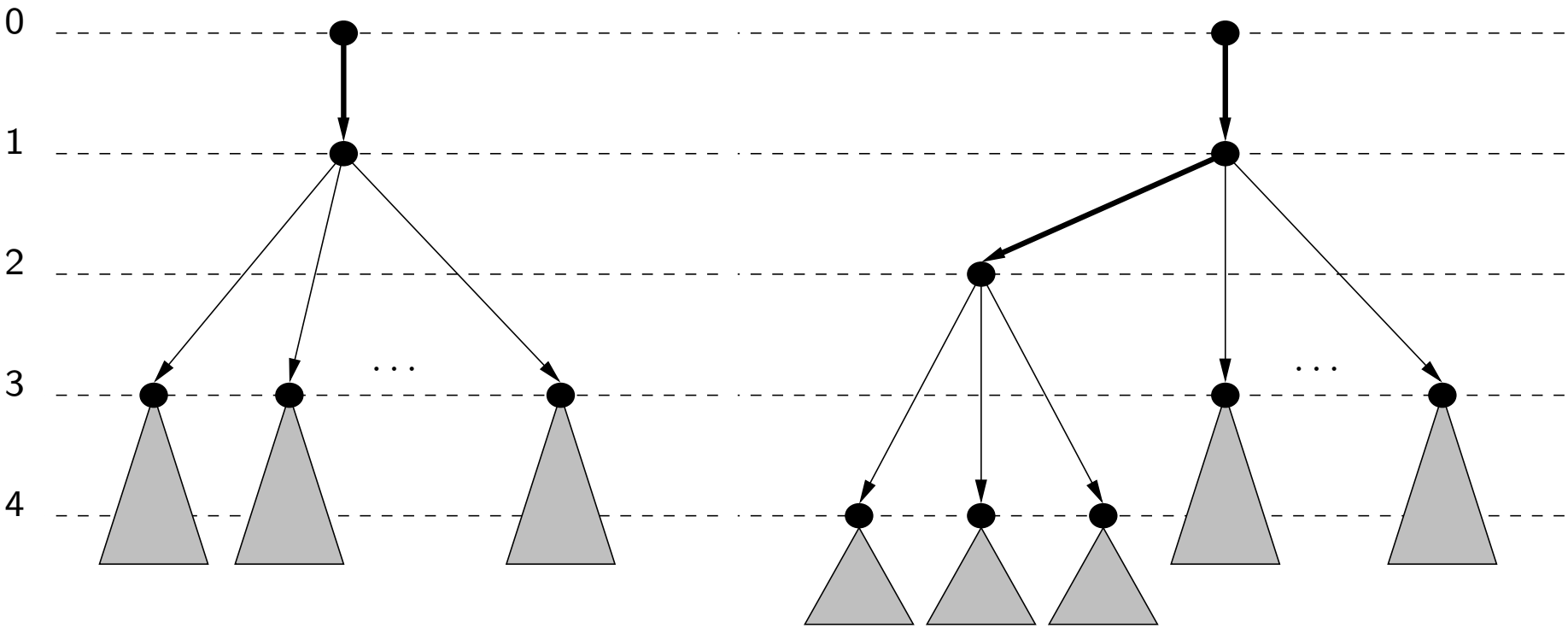
Idea

Assign the slack times as late as possible

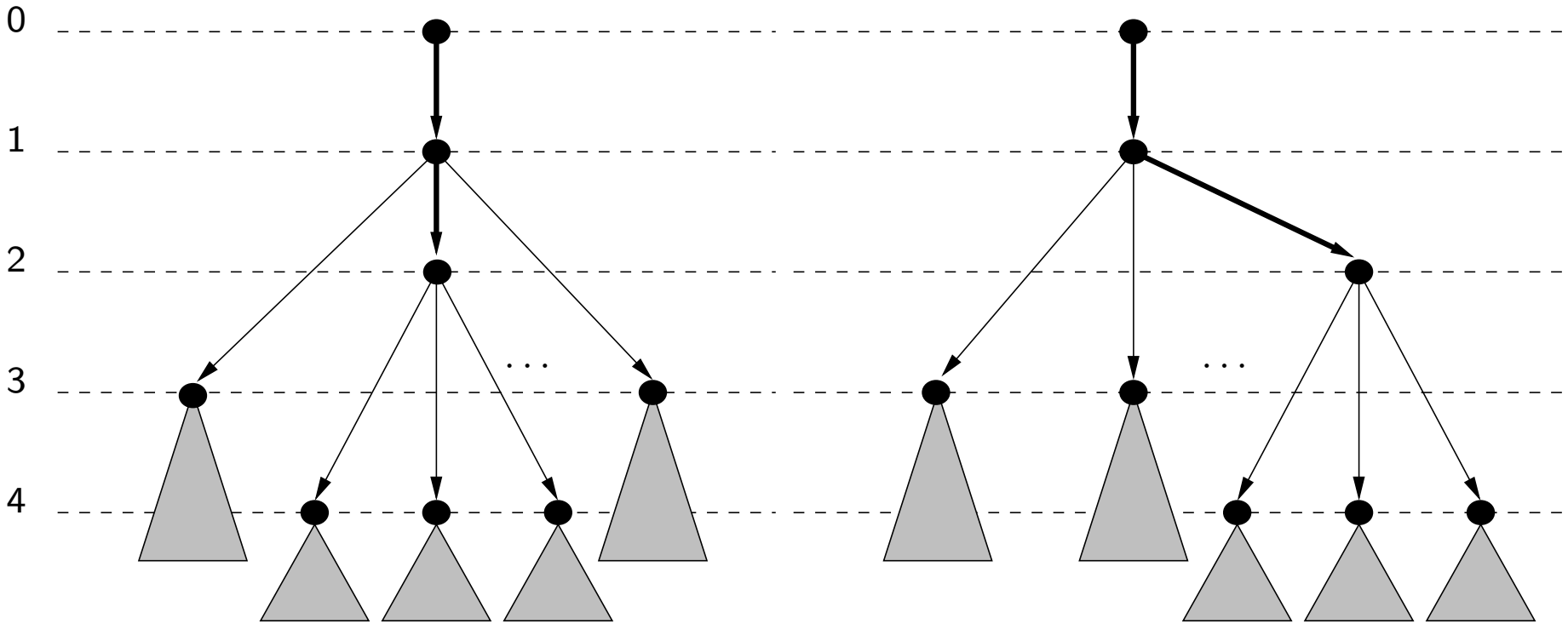
Example for $\Delta = 2$ and $L(a) = 1$ for each $a \in \mathcal{A}$



Example for $\Delta = 2$ and $L(a) = 1$ for each $a \in \mathcal{A}$



Example for $\Delta = 2$ and $L(a) = 1$ for each $a \in \mathcal{A}$



We can evaluate the cost of assigning a slack time to an edge by dynamic programming approaches

Computational complexity

We defined a pseudo-polynomial time algorithm which optimally solves $\mathcal{RTT}$ in $O(\Delta^2 n)$ time

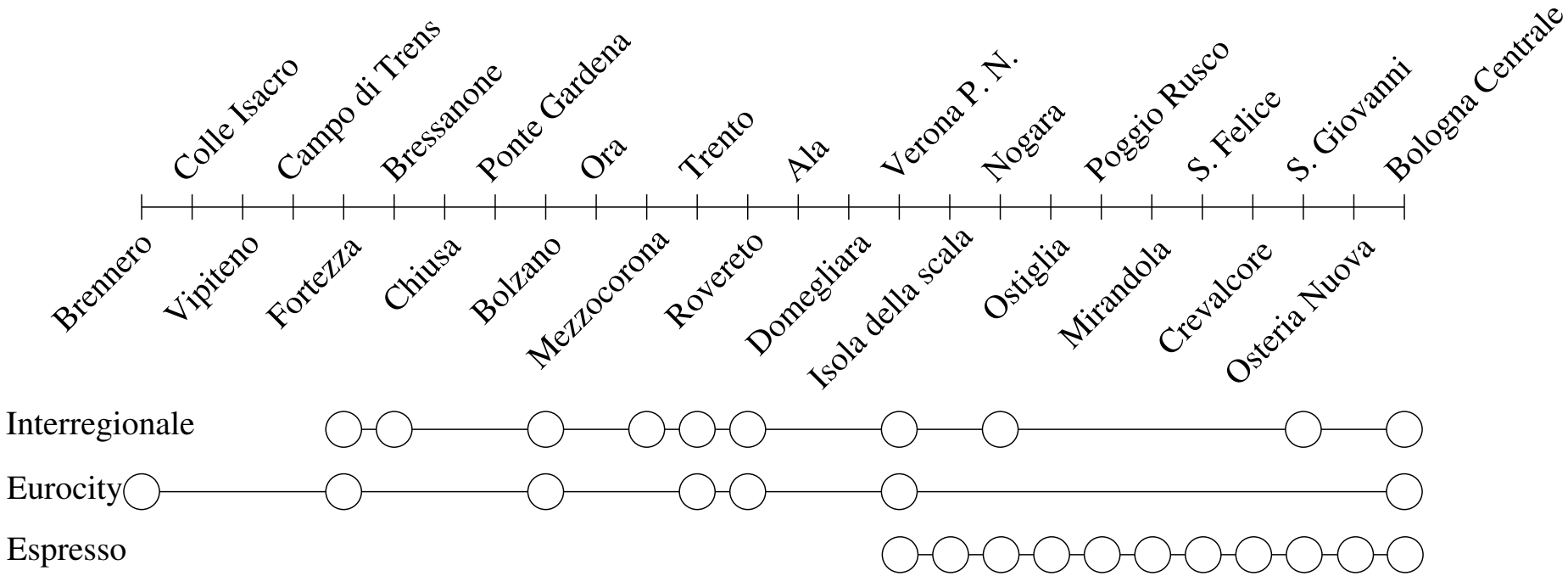
Outline

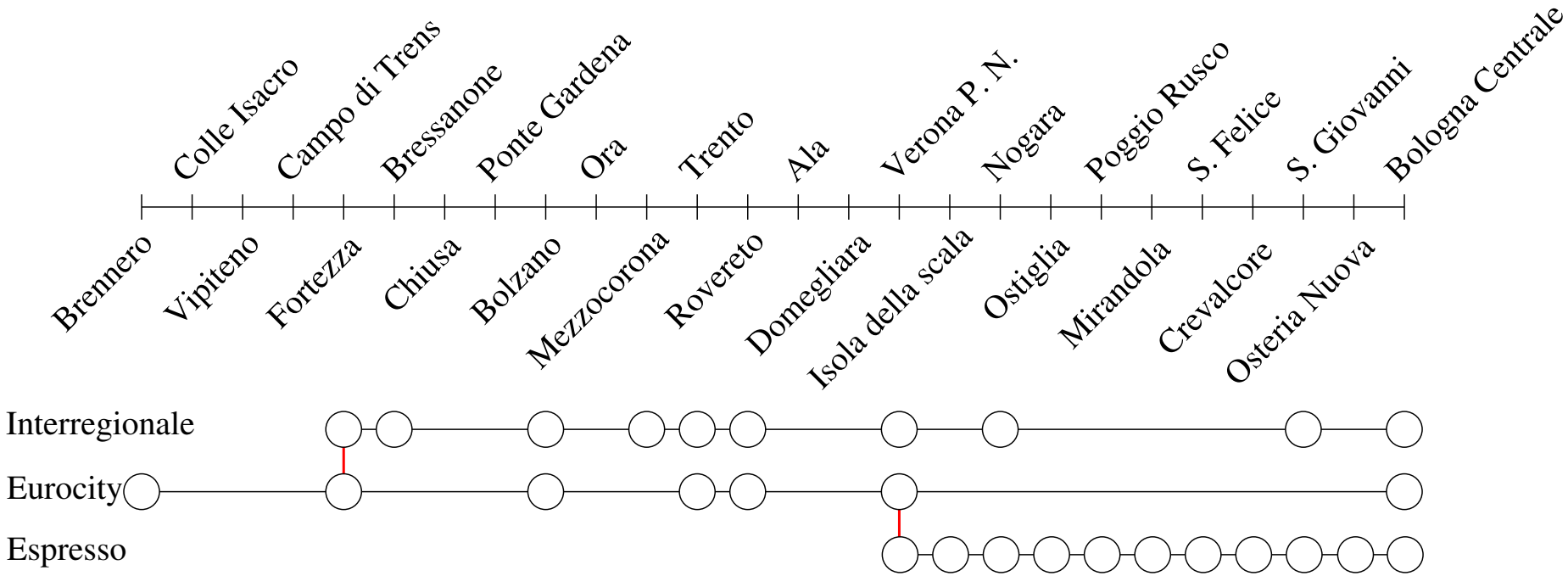
We evaluate the algorithms on two data sets:

- Italian Single line corridors
- Randomly generated data

Single line corridors

- A corridor is a set of subsequent stations served by many trains of different type
- In practice (and intuitively) slow trains wait for faster trains to allow passengers to change from one train to another
- We require that changes between trains are only those connecting a fast train to the starting event of a slow train


Colle Isacro
Campo di Trens
Bressanone
Ponte Gardena
Ora
Trento
Ala
Verona P. N.
Nogara
Poggio Rusco
S. Felice
S. Giovanni
Bologna Centrale
Brennero
Vipiteno
Fortezza
Chiusa
Bolzano
Mezzocorona
Rovereto
Domegliara
Isola della scala
Ostiglia
Mirandola
Crevalcore
Osteria Nuova
Interregionale
Eurocity
Espresso


Colle Isacro
Campo di Trens
Bressanone
Ponte Gardena
Ora
Trento
Ala
Verona P. N.
Nogara
Poggio Rusco
S. Felice
S. Giovanni
Bologna Centrale
Brennero
Vipiteno
Fortezza
Chiusa
Bolzano
Mezzocorona
Rovereto
Domegliara
Isola della scala
Ostiglia
Mirandola
Crevalcore
Osteria Nuova
Interregionale
Eurocity
Espresso

Randomly generated data

- Each tree contains 1000/3000/5000 nodes
- Each tree is generated starting from a single node and then by linking a new generated node to an existing one extracted uniformly at random.

Outline

Corridor	Line	N. of Stations	N. of Trains
BrBo	Brennero–Bologna	48	68
MdMi	Modane–Milano	54	291
BzVr	Bolzano–Verona	27	65
PzBo	Piacenza–Bologna	17	25

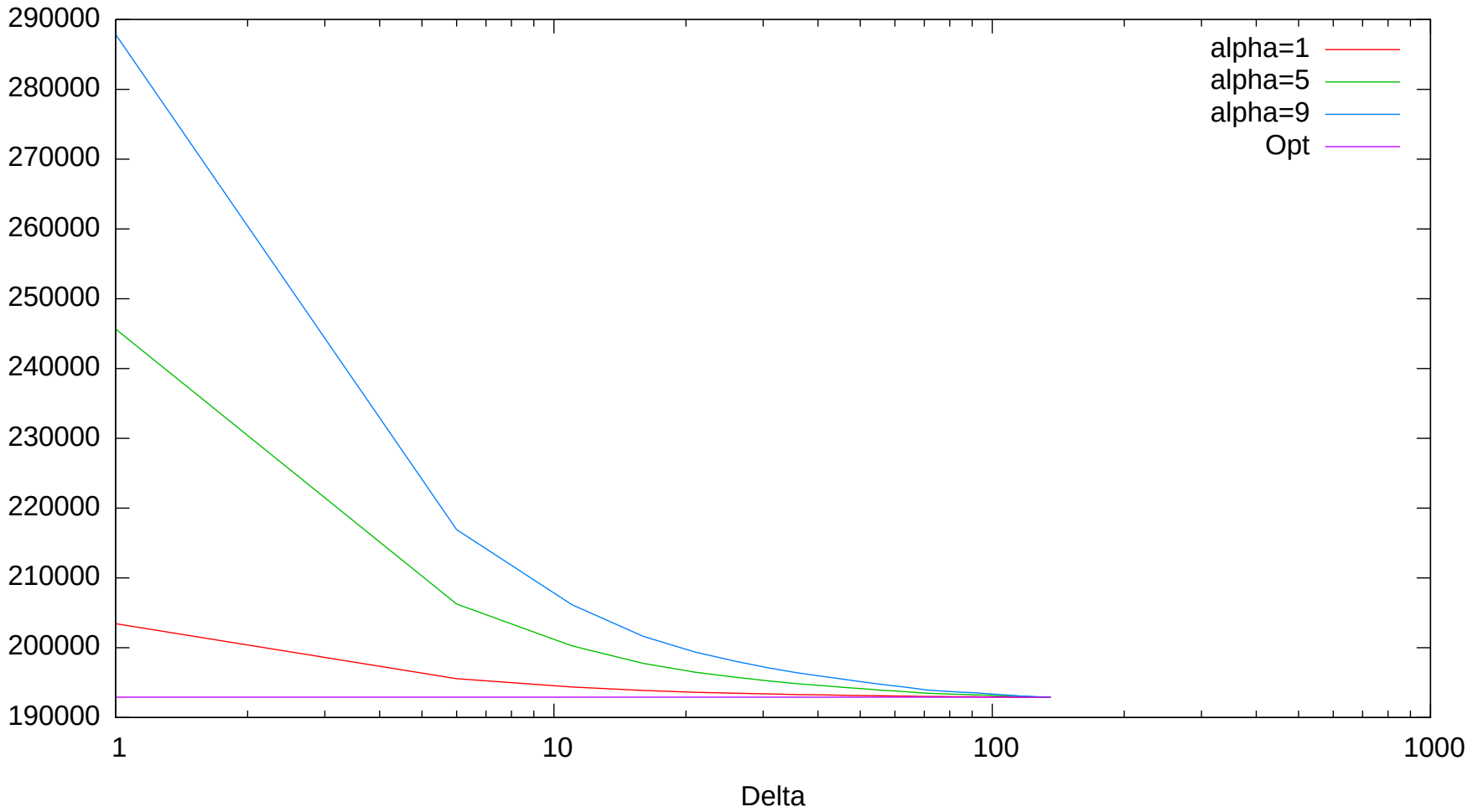
Table: Real world data used in the experiments.

Corridor	N. of Nodes	Avg Activity Time
BrBo	1103	9
MdMi	4358	8
BzVr	648	5
PzBo	163	10
Random1000	1000	9
Random3000	3000	9
Random5000	5000	9

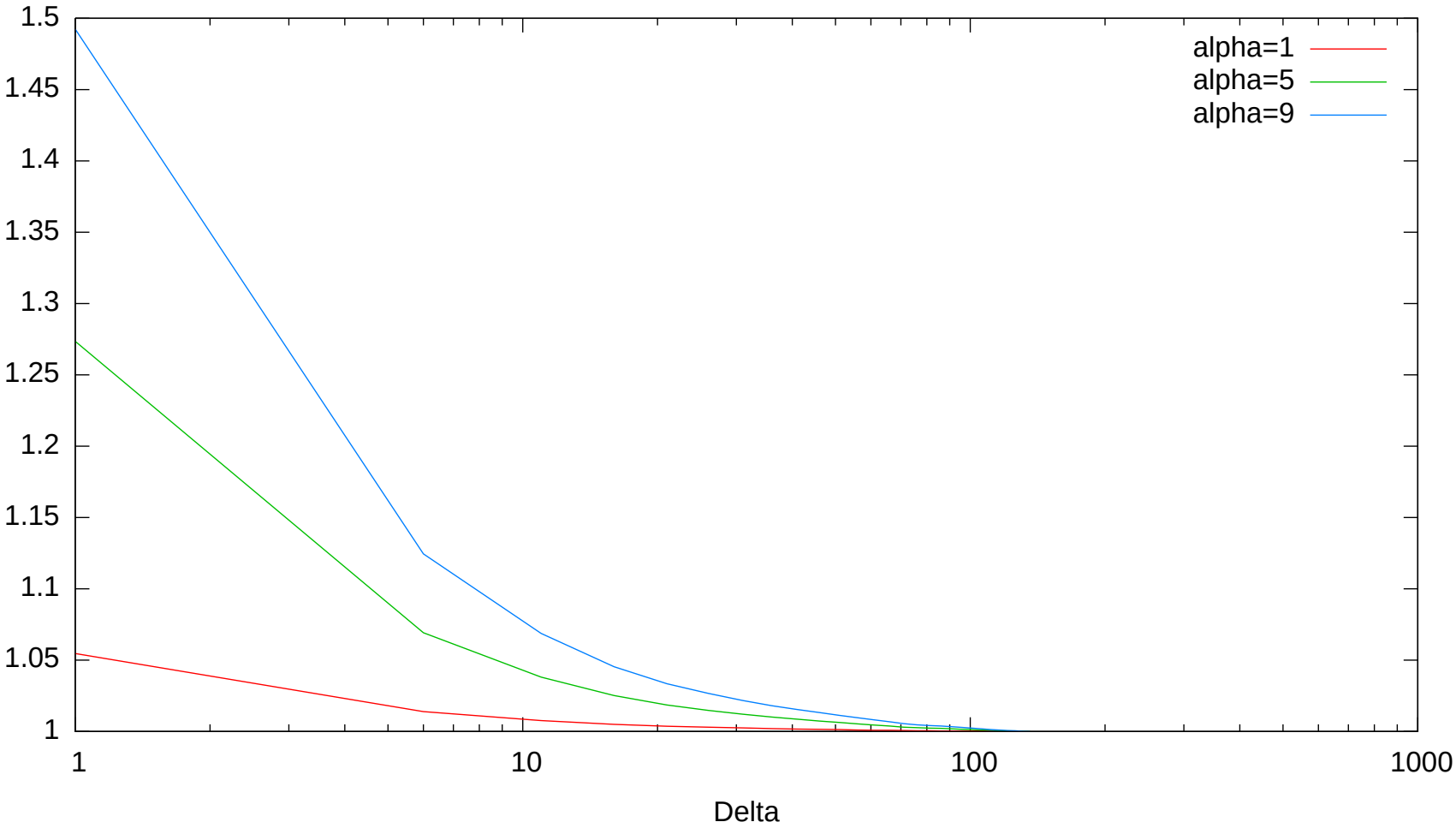
Table: Sizes of the trees.

$\alpha \in \{1, 5, 9, 13, 17\}$

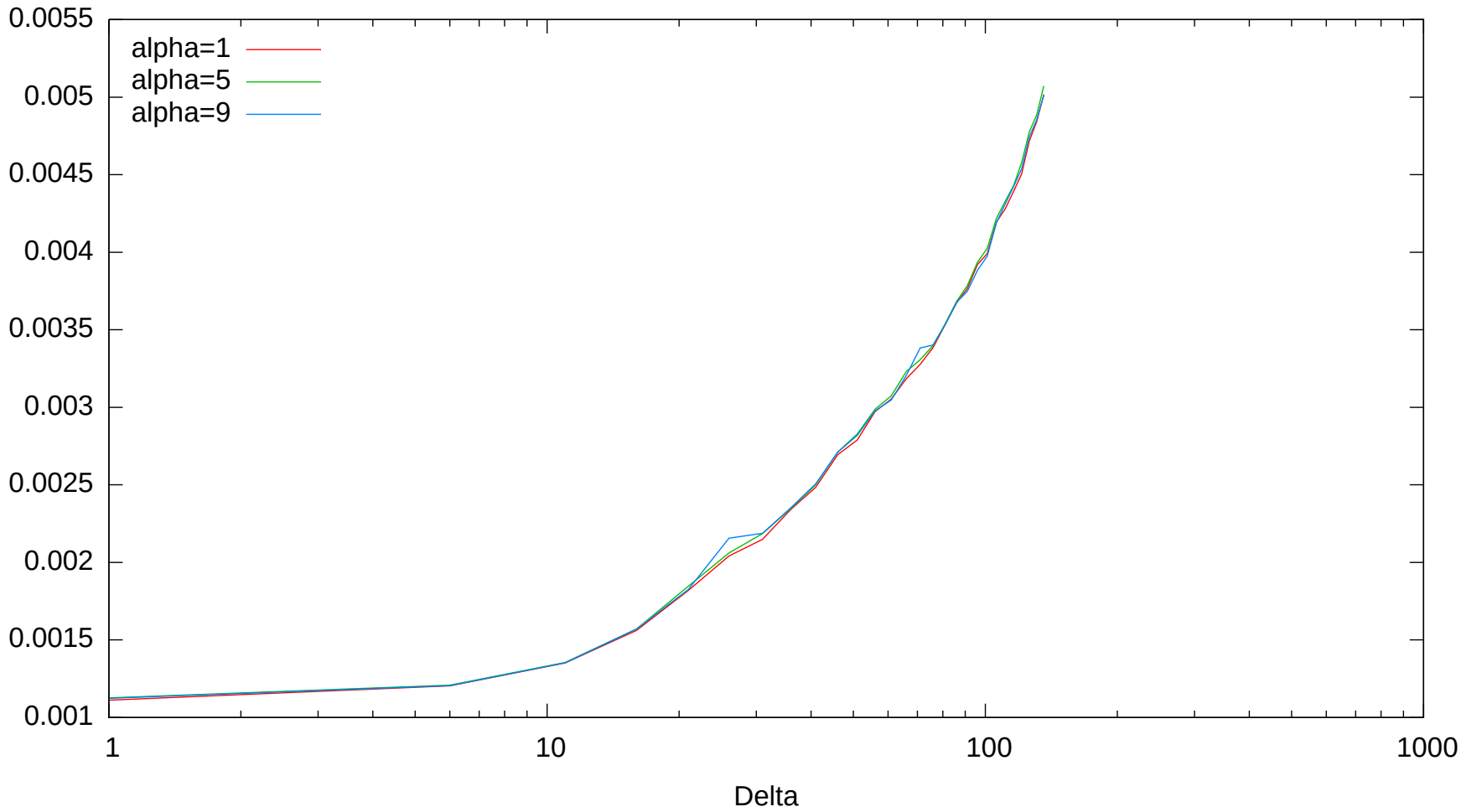
BrBo Objective function



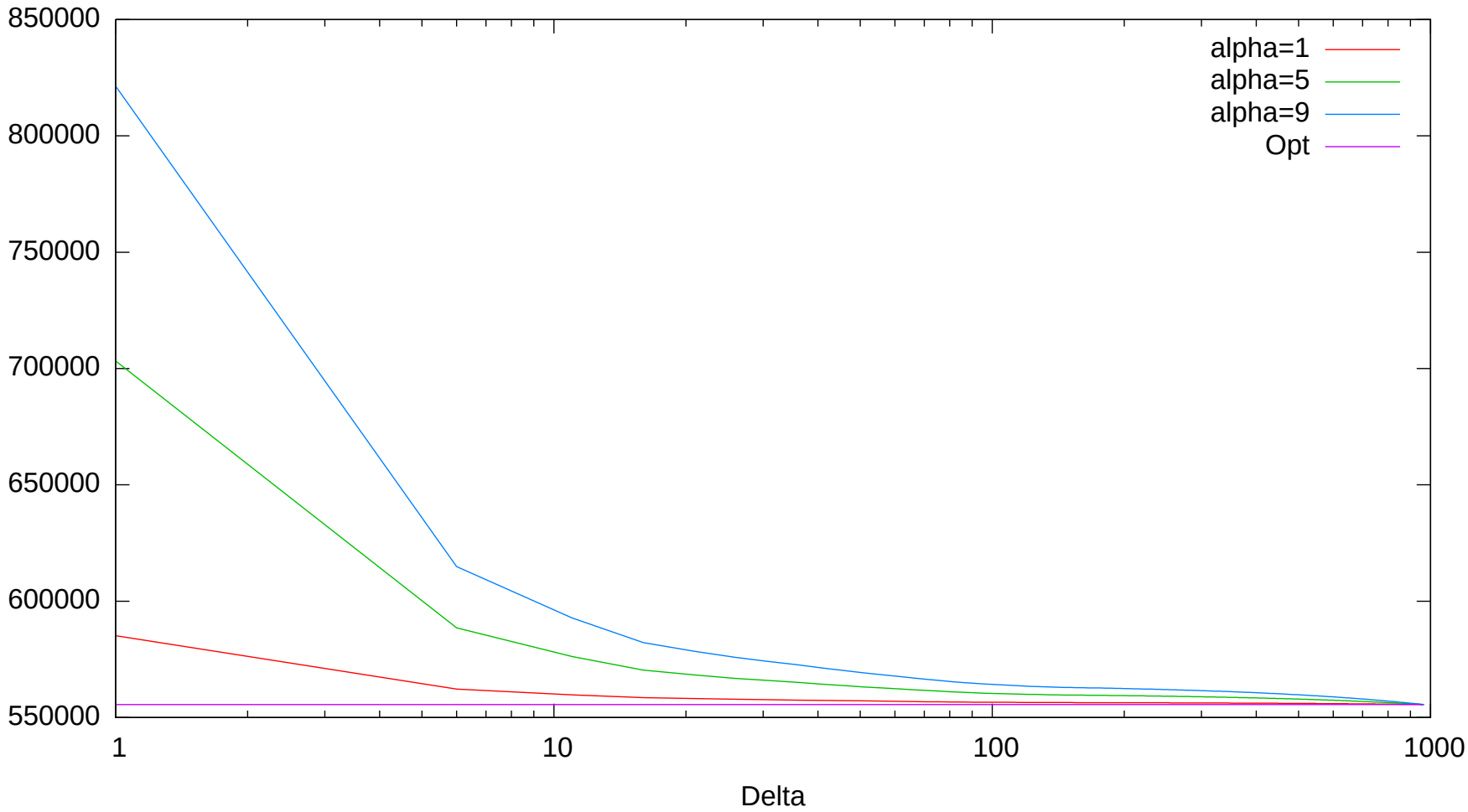
BrBo Price of robustness



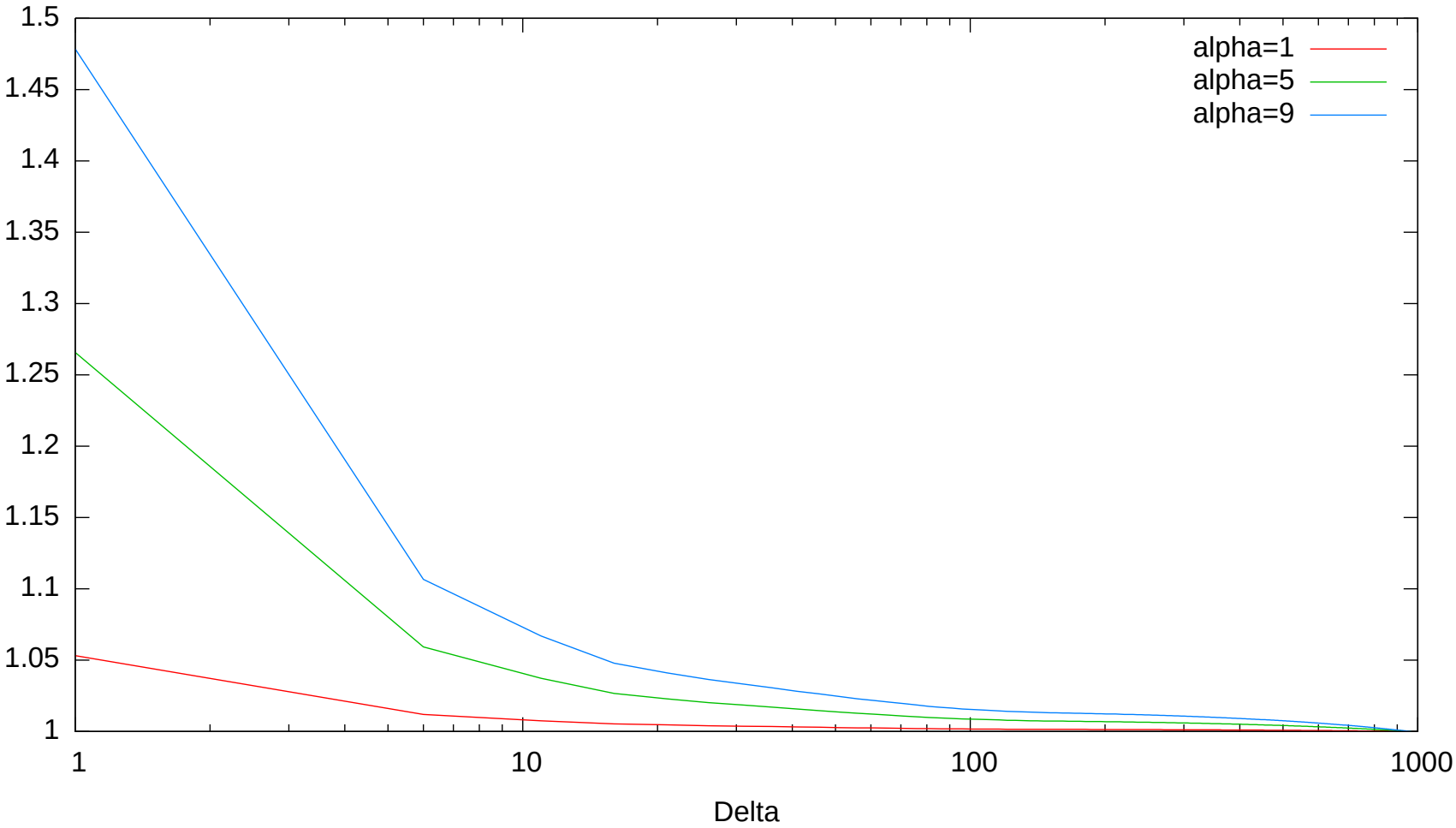
BrBo Computational time



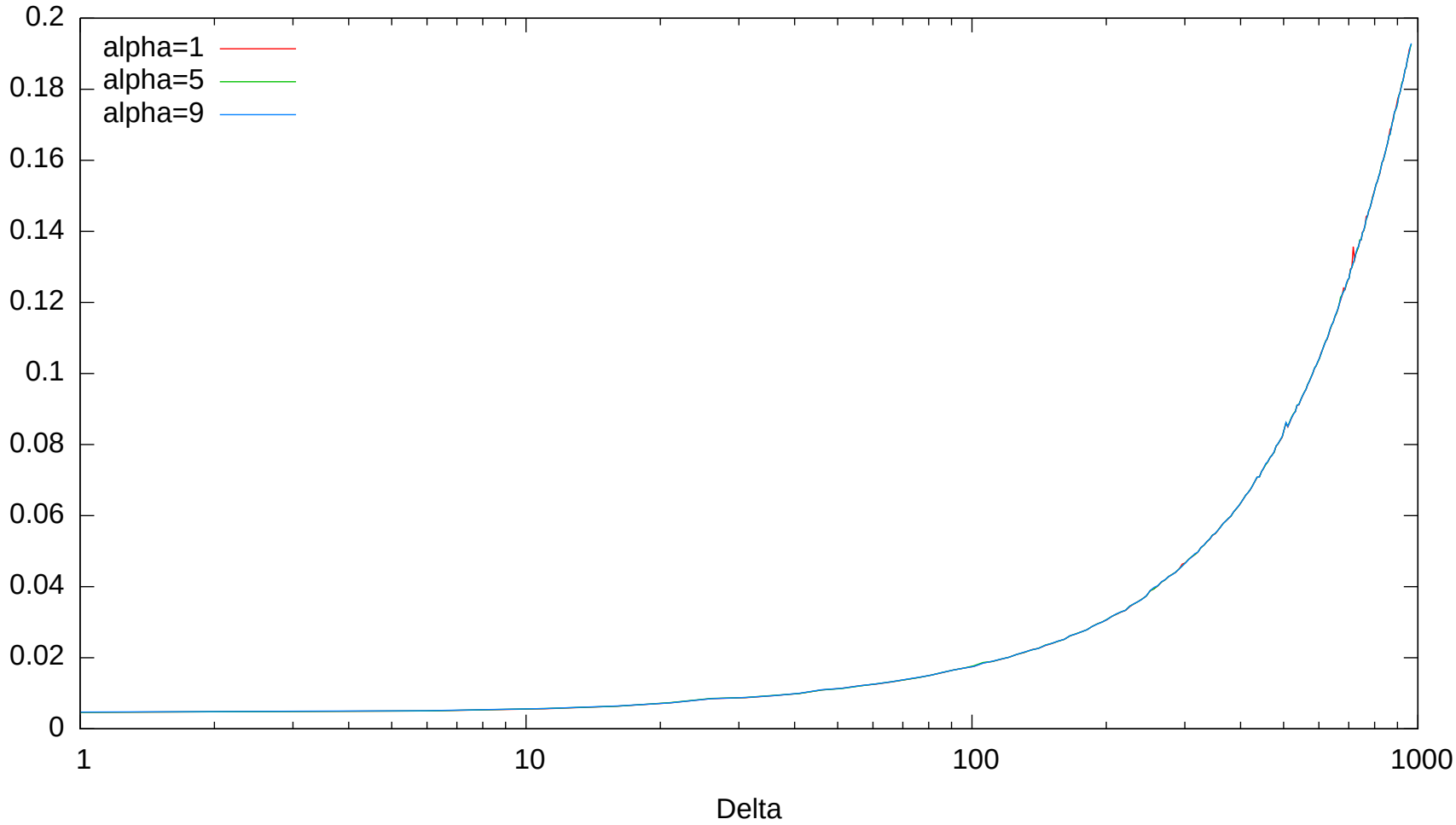
MdMi Objective function



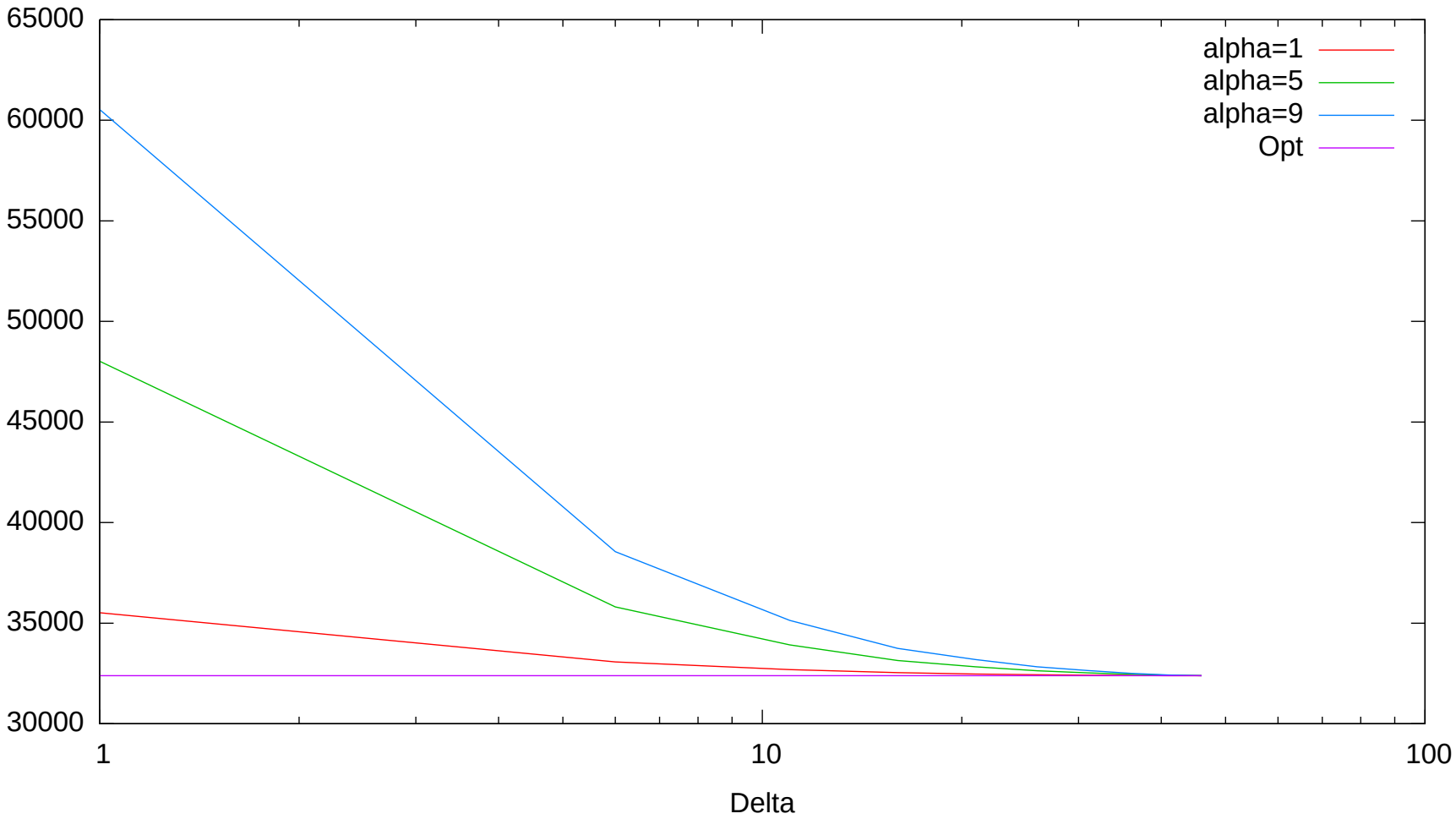
MdMi Price of robustness



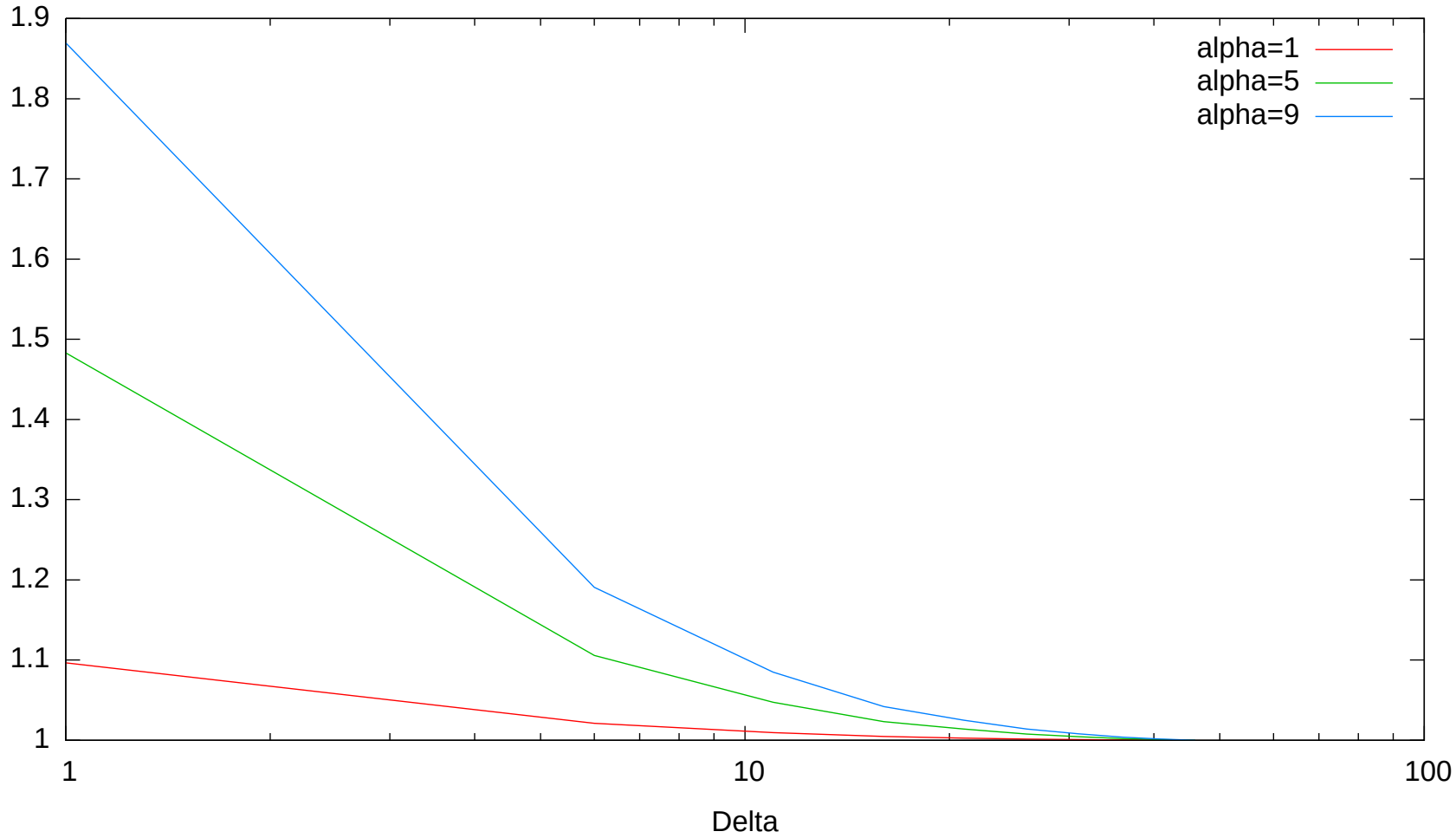
MdMi Computational time



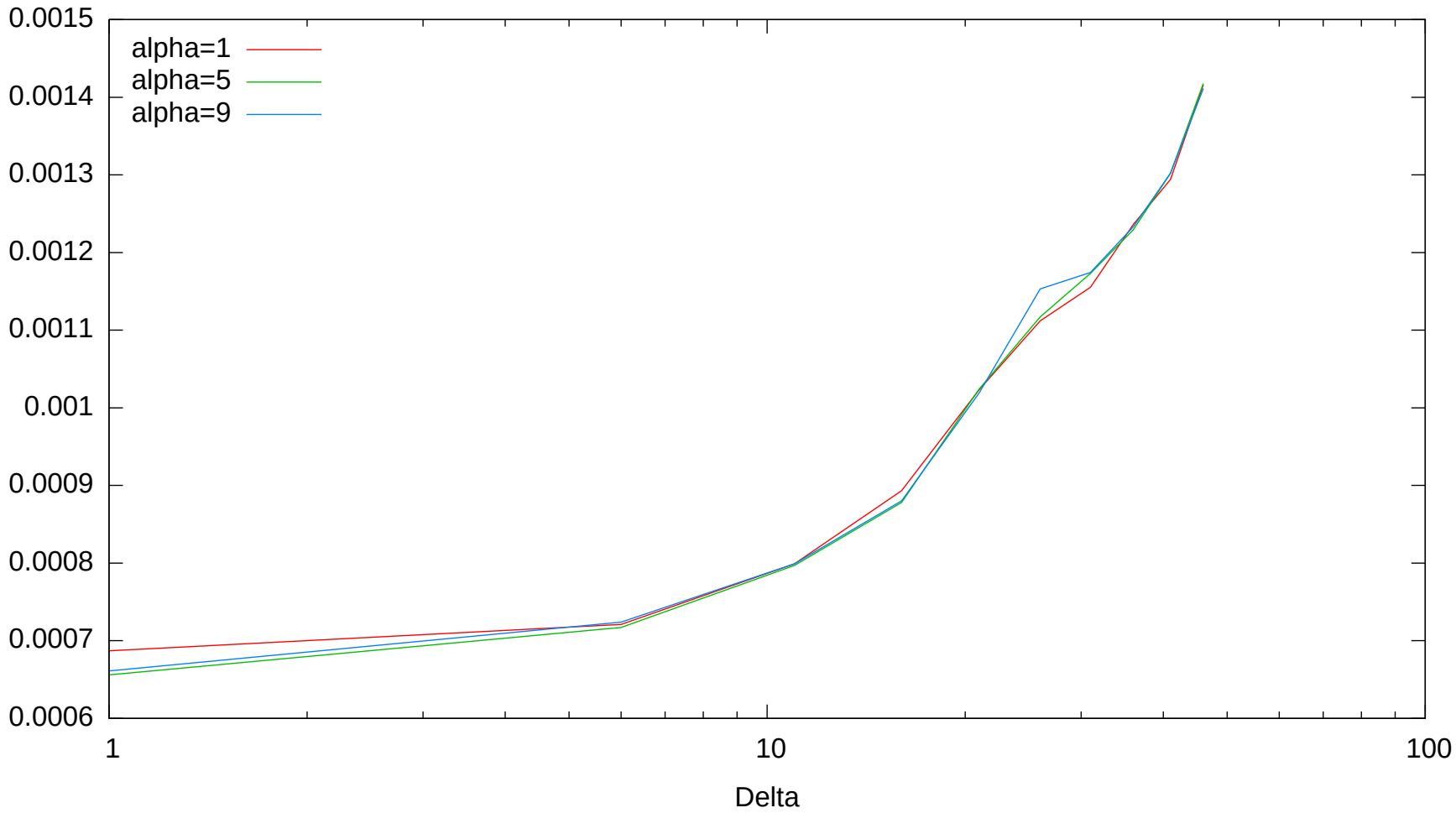
BzVr Objective function



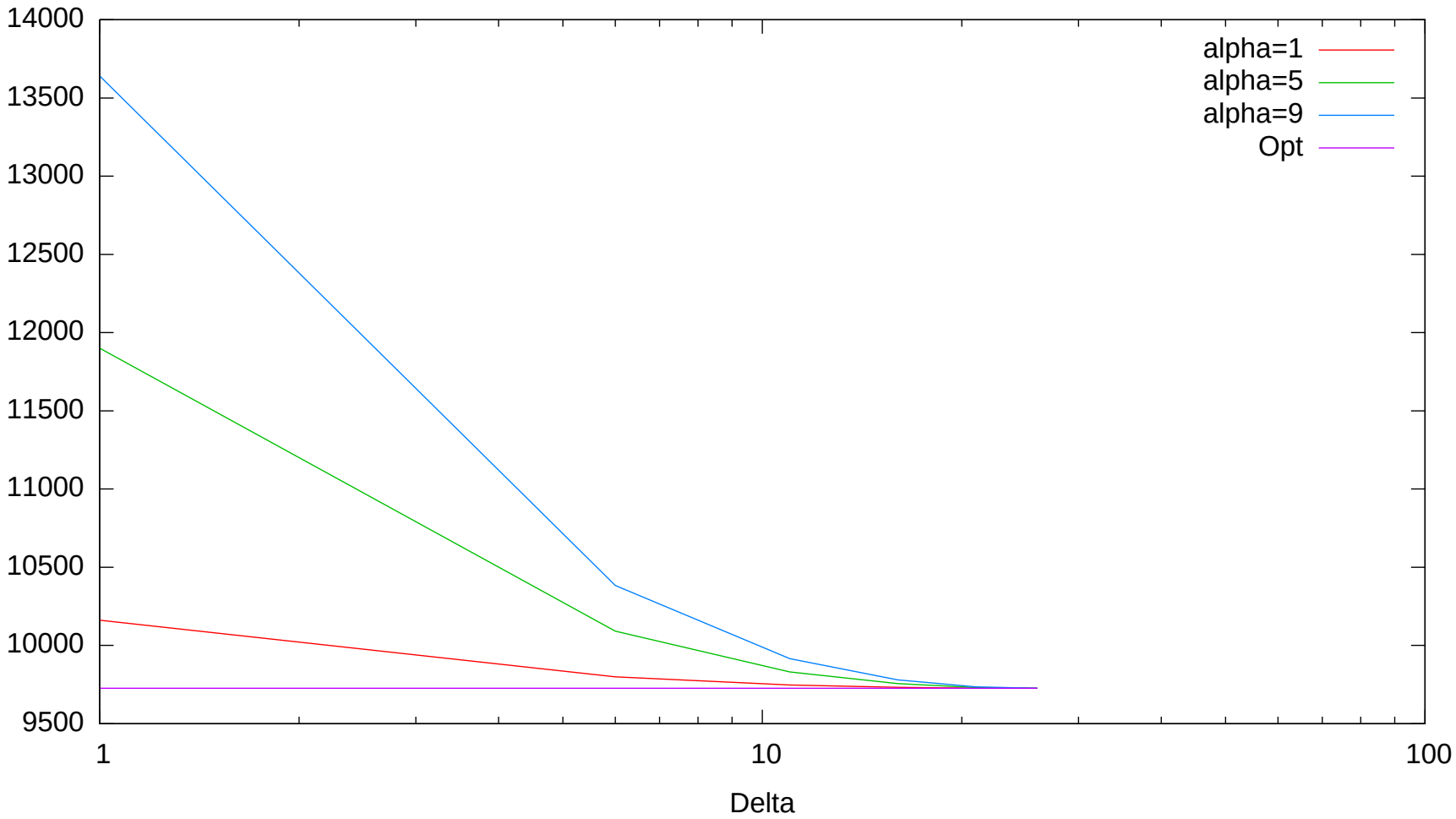
BzVr Price of robustness



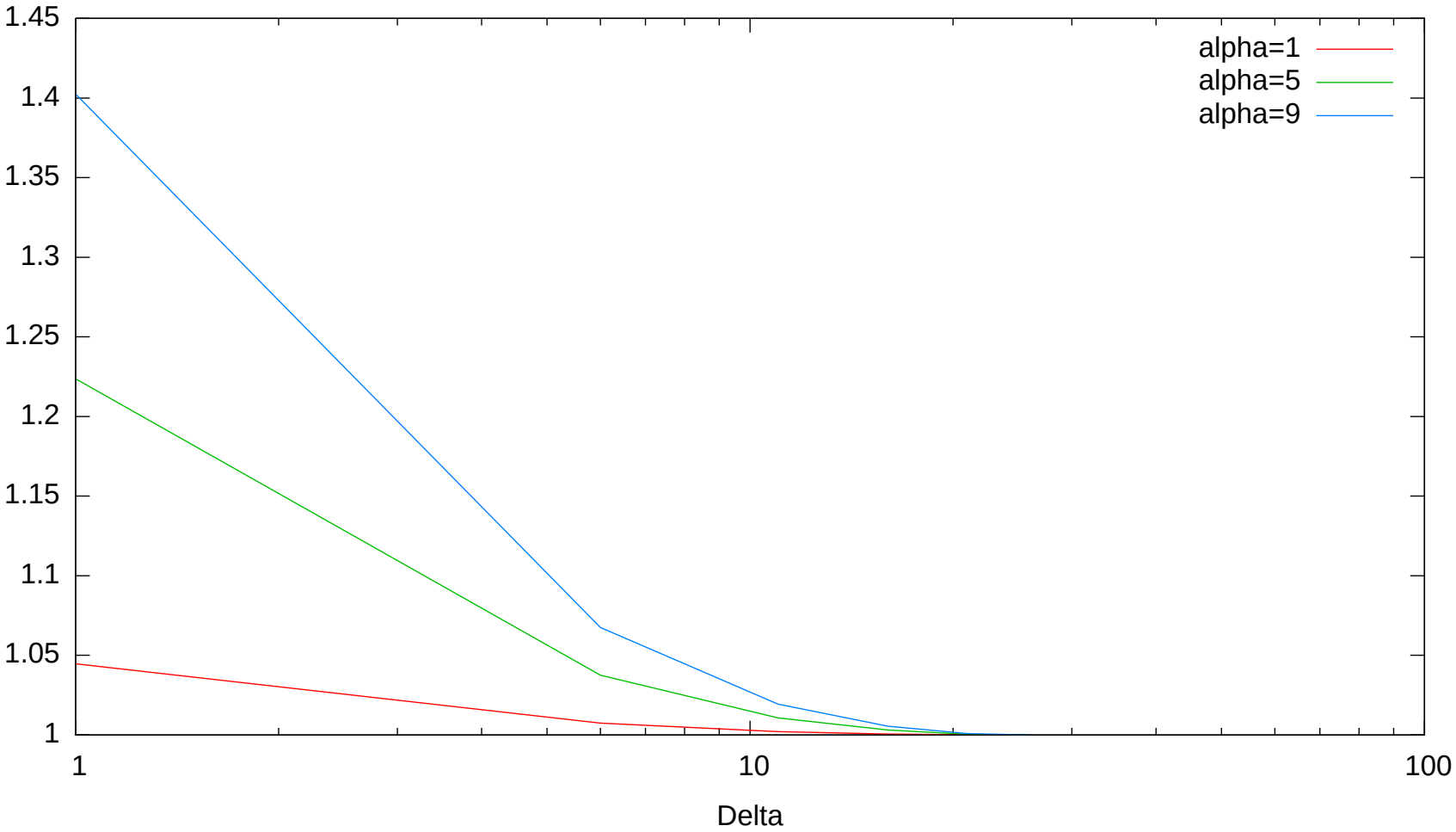
BzVr Computational time



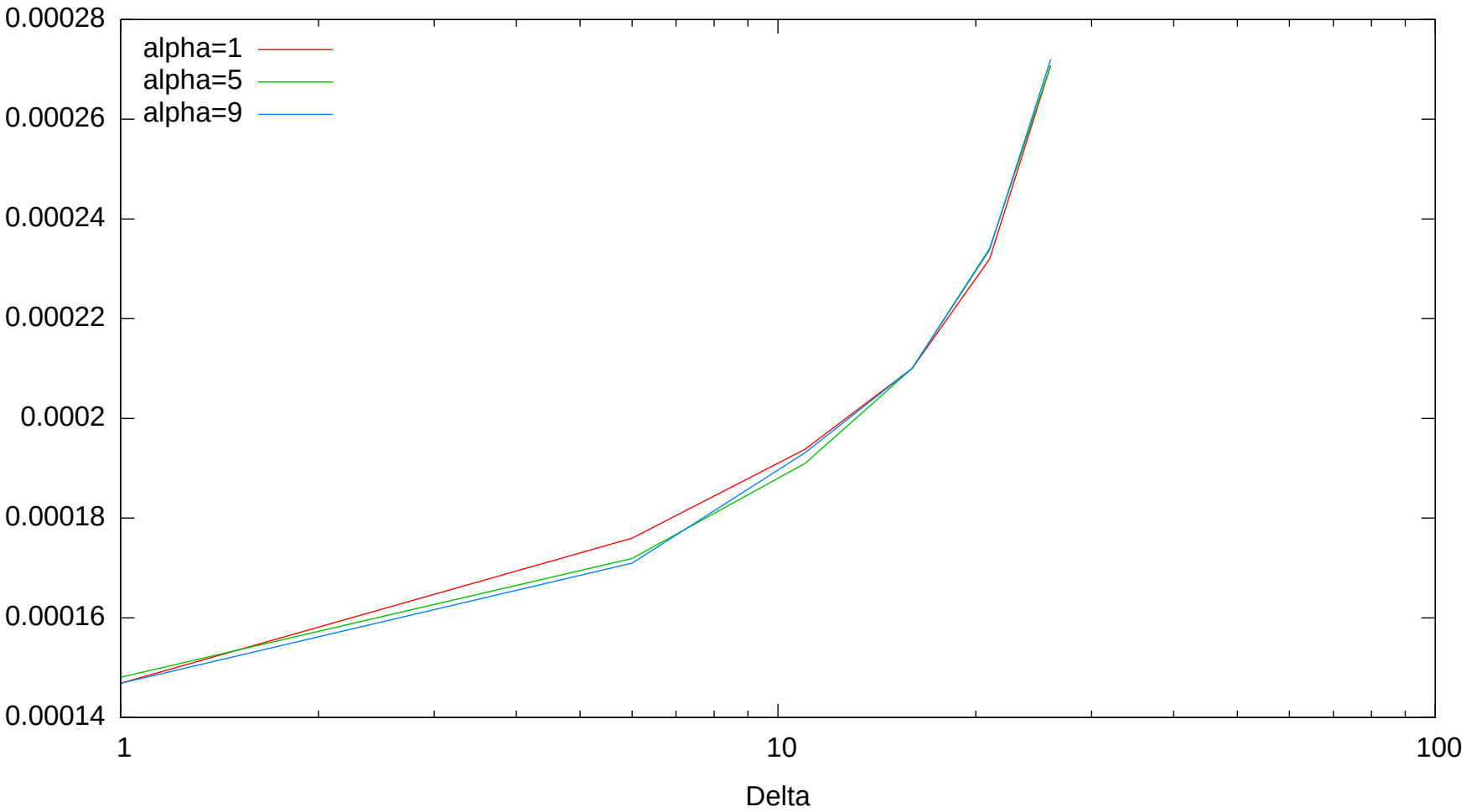
PzBo Objective function



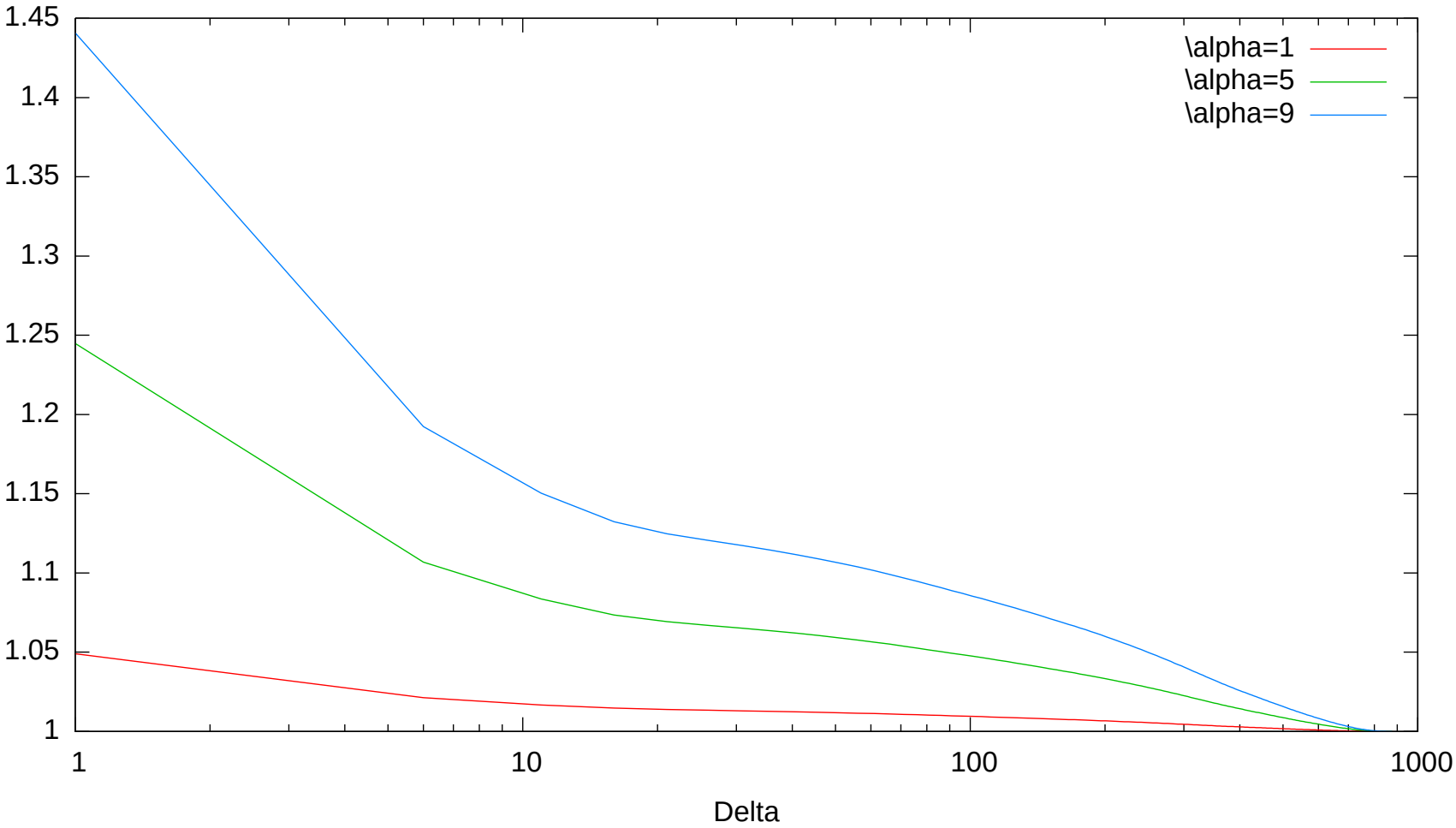
PzBo Price of robustness



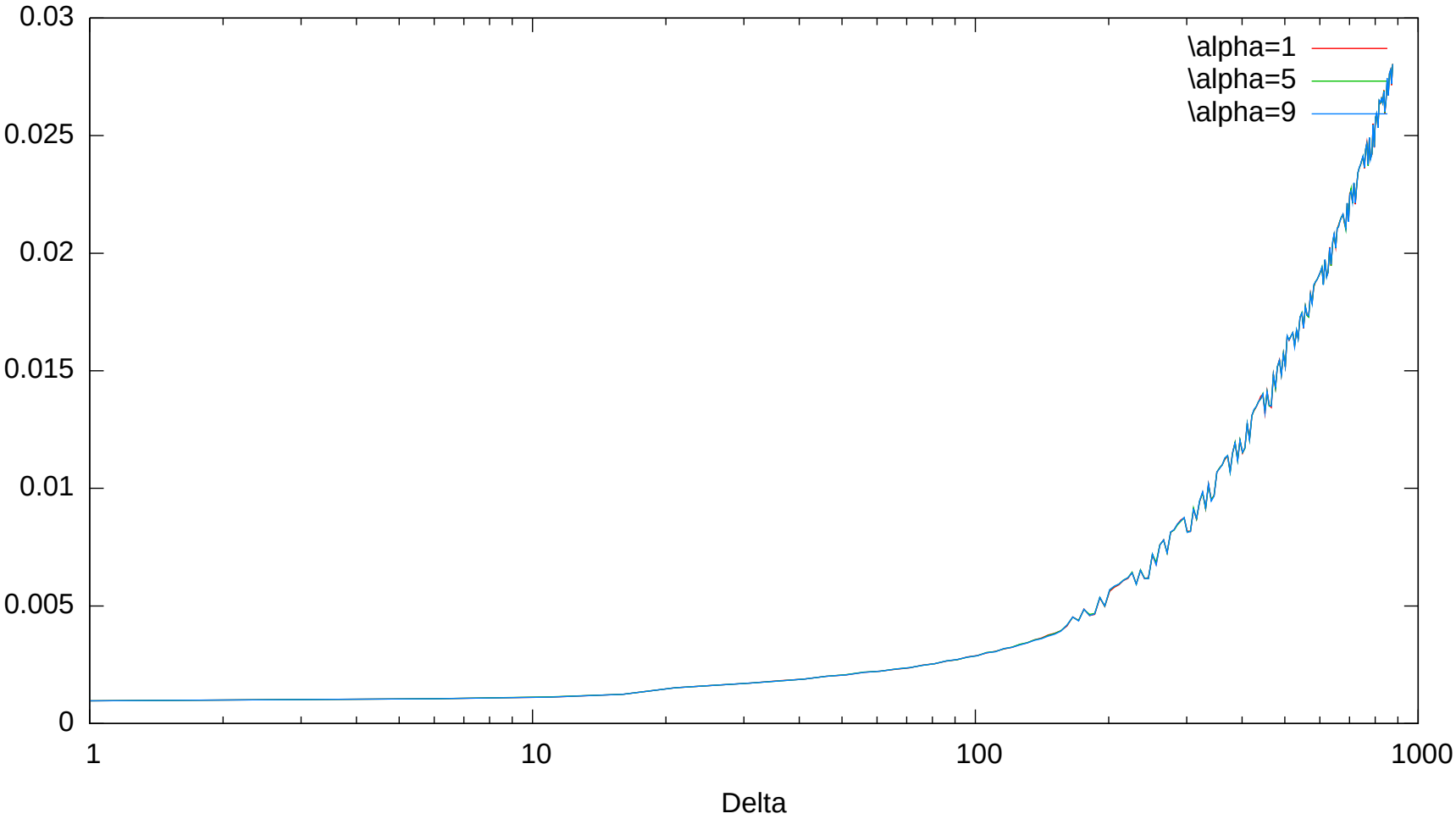
PzBo Computational time



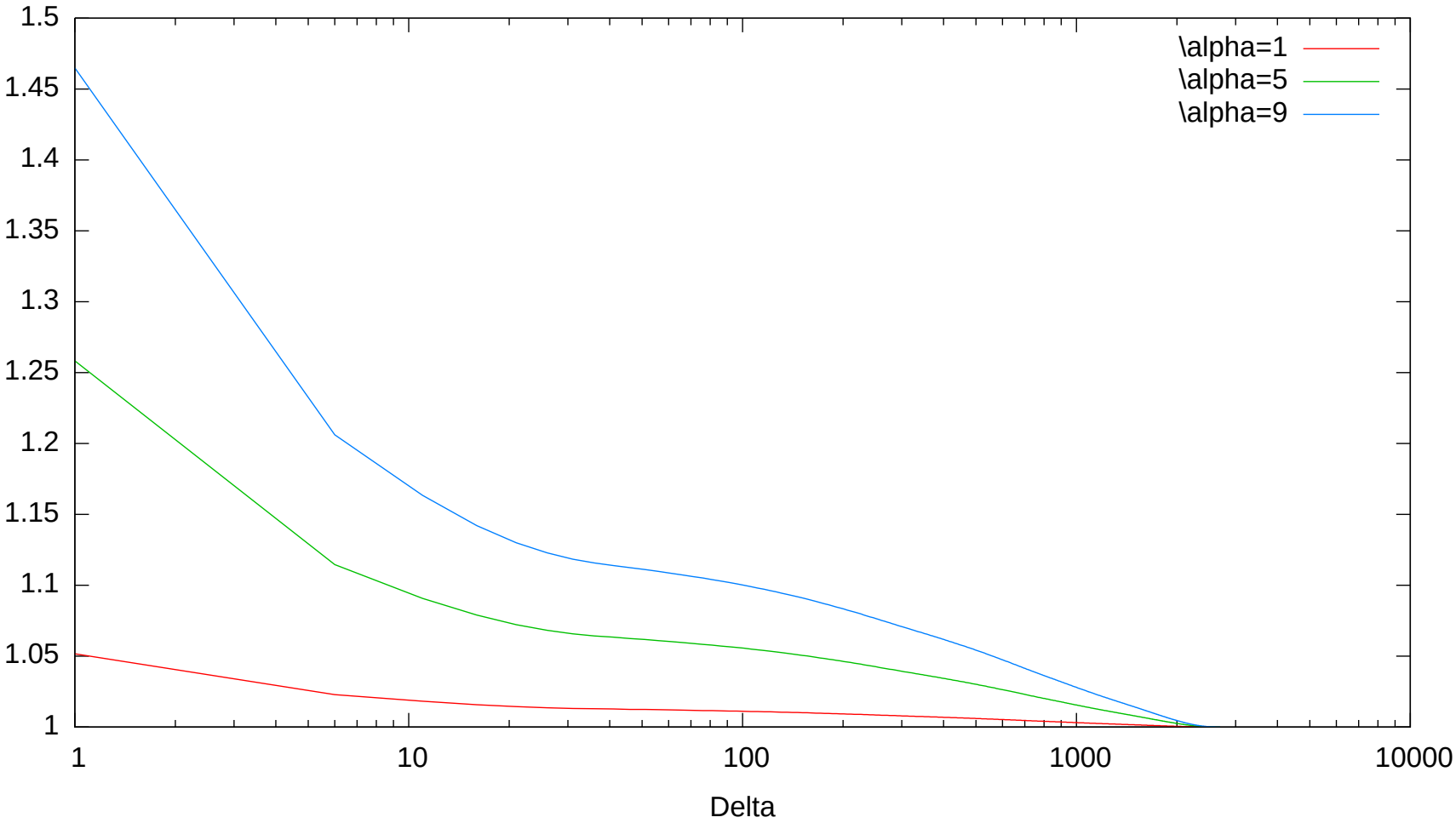
Random Graphs with 1000 nodes – Price of robustness



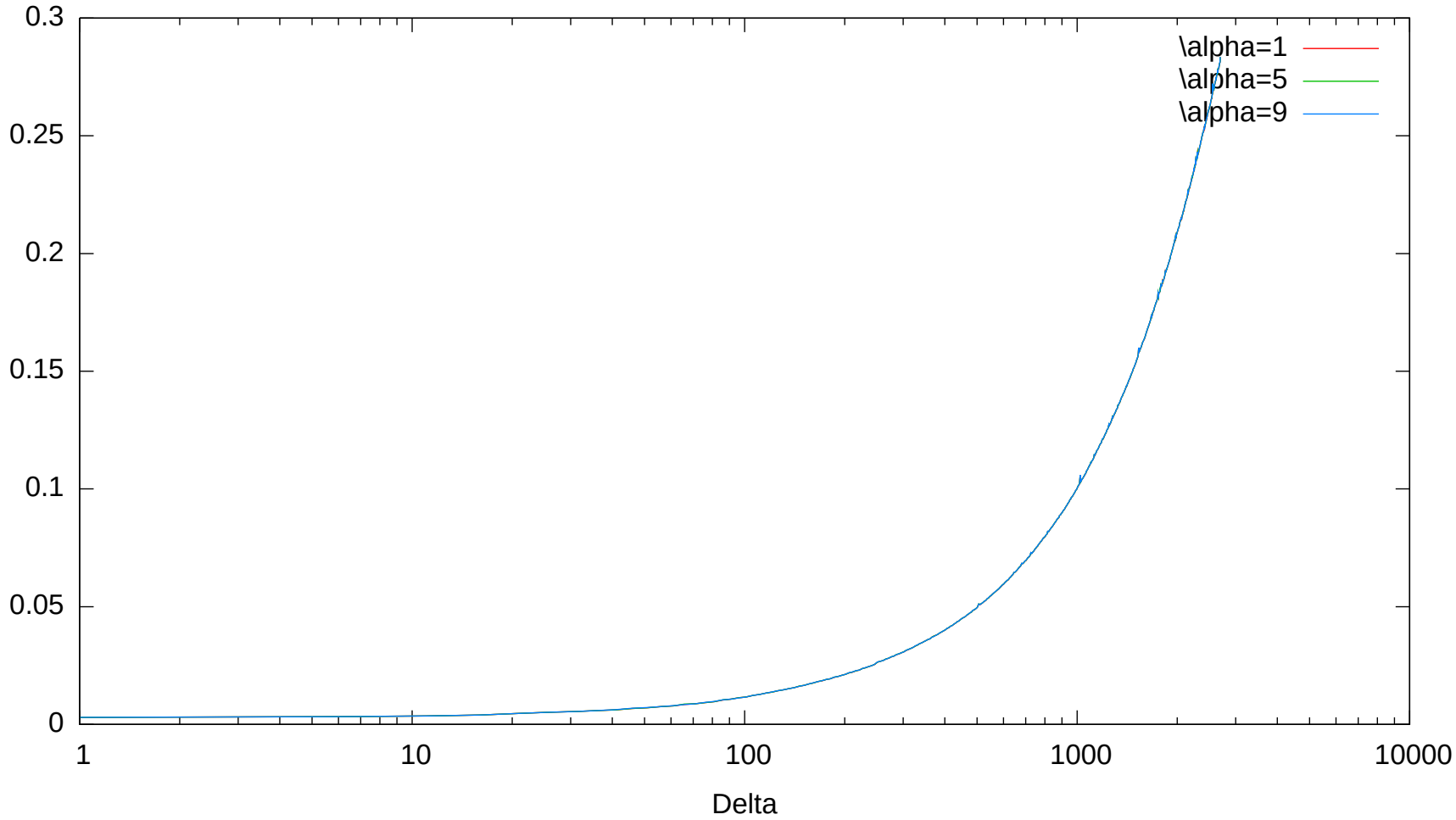
Random Graphs with 1000 nodes – time



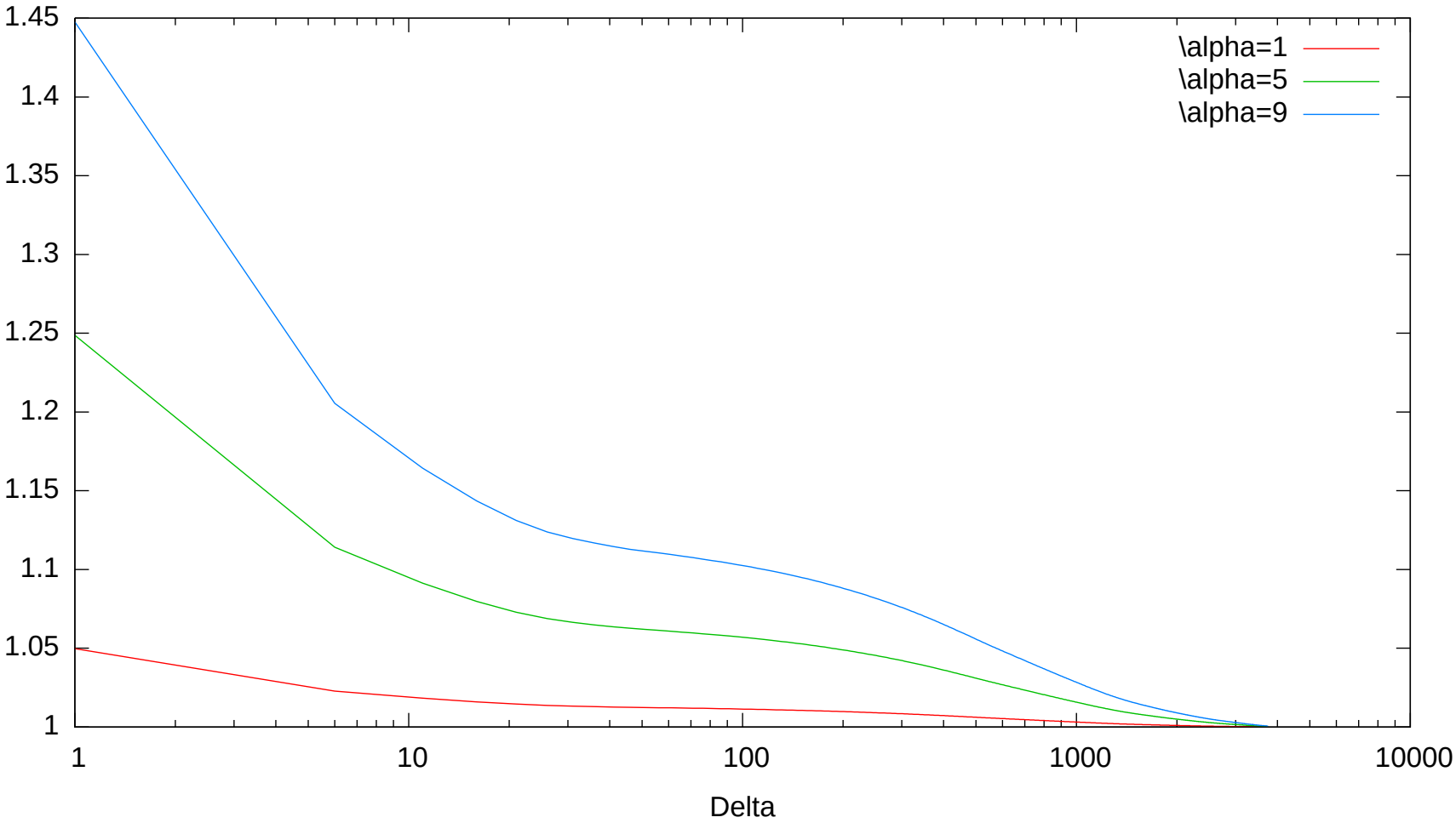
Random Graphs with 3000 nodes – Price of robustness



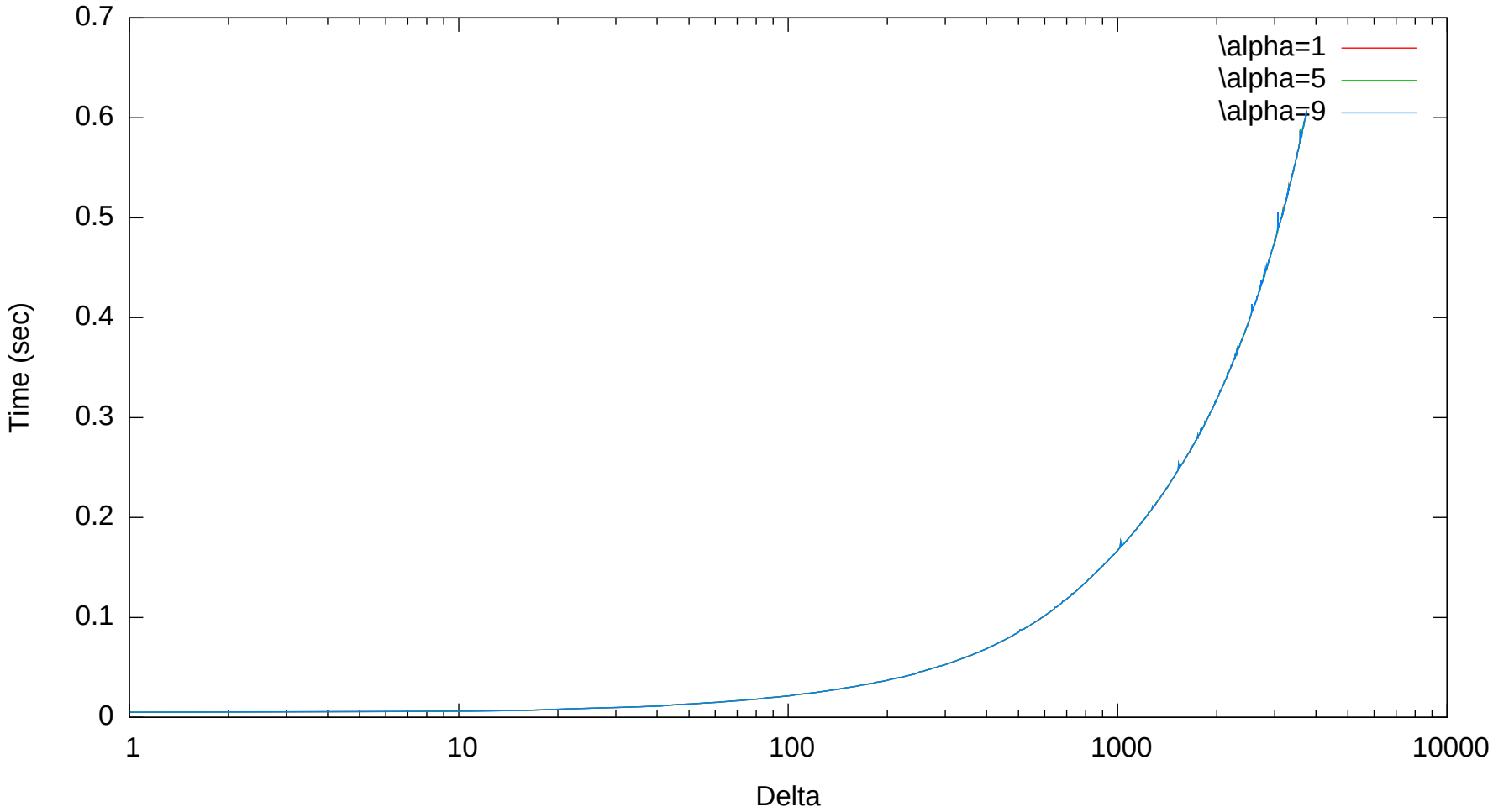
Random Graphs with 3000 nodes – time



Random Graphs with 5000 nodes – Price of robustness



Random Graphs with 5000 nodes – time



Outline

- We shown that the $\mathcal{RTT}$ problem is *NP*-hard even on tree networks
- We proposed a pseudo-polynomial time algorithm
- We experimentally shown the performances of the algorithm applied on both real data and randomly generated data
- Although the problem is proved to be *NP*-hard, the obtained results show the applicability of the algorithm